SSC17-XII-02

Using CCSDS Standards to Reduce Mission Costs

Jonathan Wilmot
NASA Goddard Space Flight Center
Greenbelt, MD, 20771, USA; 301-286-2623
Jonathan.J.Wilmot@NASA.gov

ABSTRACT

NASA's open source Core Flight System (cFS) software framework has been using several Consultative Committee for Space Data Systems (CCSDS) standards since its inception. Recently developed CCSDS standards are now being applied by NASA, ESA and other organizations to streamline and automate aspects of mission development, test, and operations, speeding mission schedules and reducing mission costs. This paper will present the new CCSDS Spacecraft Onboard Interfaces Services (SOIS) Electronic Data Sheet (EDS) standards and show how they are being applied to data interfaces in the cFS software framework, tool chain, and ground systems across a range of missions at NASA. Although NASA is focusing on the cFS, it expected that these technologies are well suited for use in other system architectures and can lower costs for a wide range of both large and small satellites.

INTRODUCTION

Over the past few years, a set of international data standards has matured to the point where their application would have significant cost, schedule, and robustness benefits for mission development, deployment, and operations. The NASA cFS team is in the process of integrating these standards into the cFS architecture and tool chain to further the goal of reducing the cost and schedule for robust high quality flight software. To provide background to the reader, this paper is divided into four sections discussing the software architecture, the data standards, mission use cases, and how the standards are being used in the architecture and development process.

INTRODUCTION TO cFS ARCHITECTURE

NASA's open source Core Flight System[1] (cFS) is a software framework and suite of common flight applications based on plug and play concepts built on a layered architecture. The architecture's well defined layers provides portability across a range of hardware platforms, operating systems and communications services. At the heart of the cFS is a publish/subscribe message service that decouples applications and provides for runtime application message exchange.

The cFS consists of a small core framework, the core Flight Executive (cFE) and a suite of several common flight applications. Due to its small memory footprint, under 1MB with FreeRTOS, the cFS is well suited to resource constrained processors. Each cFS application is a self-contained component that may be loaded, started, stopped, and unloaded at runtime using the cFE services. Messaging to other cFS components is completely location agnostic supporting multi-core, distributed, and partitioned architectures. A component can even be moved from one processor to another at runtime and the system will dynamically reconnect the message exchanges.

While the cFS provides a significant amount of flight ready code out of the box, an additional benefit of cFS is the ability to create a user library of reusable mission applications. Organizations can create a mission app store that includes their own components and those available from the wider cFS community. Early in the project development process a baseline working system can be quickly generated by selecting one of the supported operating systems, configuring the cFE and adding components from the library as needed. Once the GNU code development tools are installed on a development computer, getting a baseline cFS system up and running on x86 Linux from the Sourceforge repository takes just a few minutes. From this baseline developers focus on the mission applications as shown in yellow below.

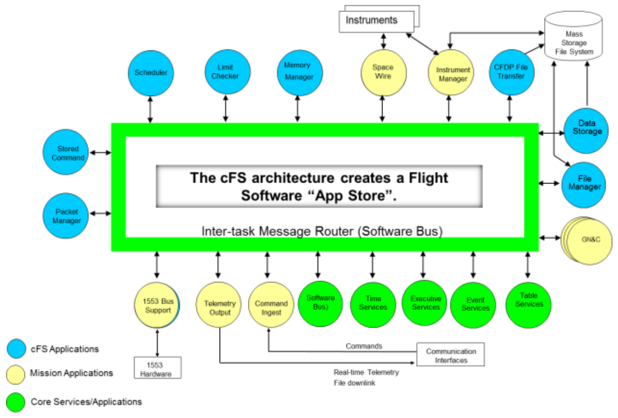

Figure 1: cFS Component Diagram

The cFS is a mature architecture and is widely used across NASA centers and is currently operating on multiple flight missions including LRO, GPM, MMS, NICER, and is the core flight software for several upcoming spacecraft including Orion Backup Flight Computer, WFIRST, PACE, and Parker Solar Probe. The cFS is also in use by universities and companies for cubesats, and UAVs, including projects competing for the Lunar XPRIZE with orbiters, landers, and rovers.

INTRODUCTION TO CCSDS

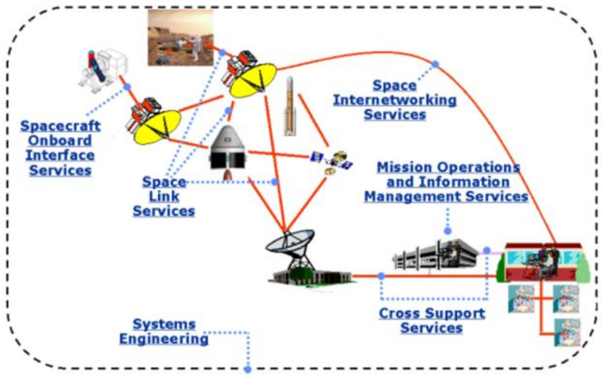
Space Internetworking Services
Spacecraft Onboard Interface Services
Mission Operations and Information Management Services
Space Link Services
Cross Support Services
Systems Engineering

Figure 3: CCSDS Organization

The Consultative Committee for Space Data Systems (CCSDS)[2] was formed in 1982 and is currently composed of eleven member agencies, twenty-eight observer agencies, and over 140 industrial associates. The CCSDS charter is to develop recommendations and standards for data and information systems to promote interoperability and cross support among space agencies, and to enable multi-agency spaceflight collaboration. All CCSDS books are license free and openly available. There are six areas within CCSDS that produce recommendations and standards for different aspects of mission operations, communications and architecture. Many of these standards have been adopted and are in use. This paper will focus on the standards work of the Spacecraft Onboard Interface Services (SOIS) Area.

Onboard Interface Services (SOIS) Area

SOIS is chartered to develop standard service interfaces that are provided to onboard software applications. These service interfaces are intended to isolate the flight software from the underlying hardware details and thereby increase the portability and reuse potential of the flight software. The service access points also constitute cross support interfaces. The SOIS Working Group developed a layered architecture and published a set of recommendations for the common service access points. Initial feedback from space agencies and commercial spacecraft vendors stated that although the layered architecture and service access points were common and used within their existing architectures, adopting the standards would not bring sufficient benefits to their development process and business model. They did however, see significant value in the Electronic Data Sheet (EDS) concept as an interoperability point. For system and component interoperability the EDS was seen as a way to automate several aspects of spacecraft development, operations and maintenance. With this feedback in mind the SOIS Working Group decided to focus on the EDS and associated software tools.

THE SOIS EDS

A SOIS Electronic Data Sheet (SEDS) is a layered description of a hardware or software component interface in a machine-readable format. It provides a standard to exchange mechanism for device and software interface definitions between organizations and agencies. The SEDS is intended to replace traditional interface control documents and proprietary data sheets which are necessary to determine the operation of the device and how to communicate with it. A component SEDS is an unambiguous definition of those interfaces enabling software tools to ensure consistency and completeness of information. With software tools SEDS can dramatically improve the development process by:

a) Generating human-readable documentation
b) Automatically generating software, implementing the relevant parts of the OBSW for the component
c) Automatically generating device interface simulation software for use in test or device-simulation software
d) Transforming the device functional interface into telecommands and telemetry suitable for processing by a command and data handling system on-board and on the ground
e) Capturing and exporting interface information for the spacecraft database

Use Cases for EDS: Definition Phase

Even before a mission is in development the SEDS can be used in the proposal and mission definition phases to specify the system and generate cost estimates. Several years ago the Air Force Research Lab (AFRL) developed the Space Plug and Play Avionics[5] (SPA) concept and some tools to allow mission engineers to input mission specifications: orbit, pointing requirements, instrument parameters and the tools would scan through a database of EDSs and select the hardware and software components needed to achieve the mission. This approach is very useful early in mission formulation to determine what existing components can be used and what new components need to be developed leading to a much better estimate of cost and schedule.

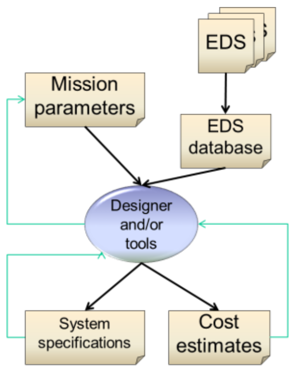

Figure 4: EDS Use in the Mission Design Phase

Although there are some fundamental differences between a SPA extensible transducer electronic data sheets (xTEDS) and a SEDS, the concepts are very similar. While a SPA EDS describes SPA compliant devices, a SEDS can describe an arbitrary software or hardware device. This difference is due to the fact that SPA was trying to achieve true run-time Play and Play while the cFS and SEDS are targeting build time integration with runtime discovery and communication systems configuration.

Use Cases for EDS: Mission Development Phase

During the development phase the EDS has several use cases that include auto-generation of flight code, test procedure, simulators, system models, and ground system databases.

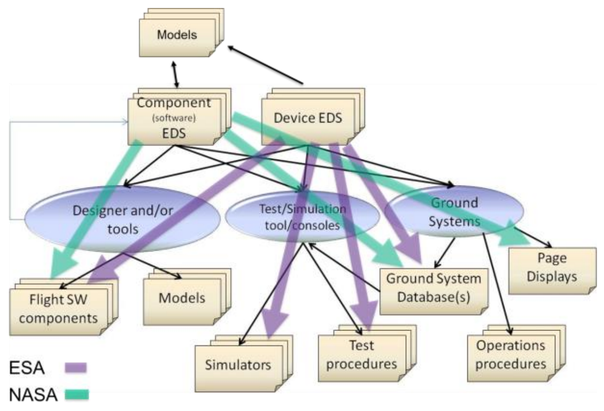

Figure 4: EDS Development Use Cases

Both ESA and NASA have developed tools to support using the SEDS during mission development.

In 2015 and 2016 the European Space Agency (ESA) funded study projects that demonstrated tools reading the SEDS for a hardware device, in this case a Jena-Optronik star tracker, and auto-generate the software device driver, a device simulator, the device command and telemetry database, and device test procedures. Those development artifacts were used to interface, control and test the real star tracker hardware in a simulated mission environment. The results of the study will be published in an upcoming conference.

NASA has been using non-standard "EDSs" for several years. Previously these data sheets had taken the form of Excel spreadsheets, and custom XML files. With the SEDS now ready for publication, the cFS team is migrating to the CCSDS standard and has demonstrated the auto-generation of flight code, ground system command/telemetry databases, and ground system displays. The migration to SEDS is on track for inclusion in the next open source release of the cFS flight code in Fall 2017.

Contents of a SOIS EDS

A SOIS Electronic Data Sheet is a set of related XML files as specified in the corresponding CCSDS schema[3] and dictionary of terms[4]

Such a datasheet can contain definitions of:

- The **interfaces** that allow two-way data interchange between layers of the OSI protocol stack.
- The **commands** and **parameters** that make up such interfaces.
- The **components** that make up the services that implement a mapping between two sets of interfaces.
- The **state machines**, **variables** and **activities** that make up such components.
- The **types**, **ranges**, **encodings** and **semantic terms** referenced by any of the above.

The intent of the schema is to be usable as an *interchange format* between a varieties of tools that need access to the information it contains. Consequently, the design of the schema has been performed according to the following principles:

- Avoid ambiguity or open areas in the specification.
- Limit supported features to those known to be used by one or more real devices.
- Limit the number of different ways it is possible to represent any given device feature.

- Minimise the amount of external validation needed to detect logically inconsistent specifications.

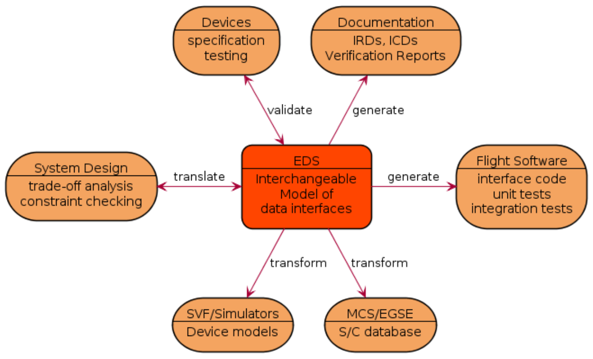

Figure 5: EDS Context Diagram

The EDS context diagram in Figure 5 shows the relationship of the EDS to the different artifacts needed during development.

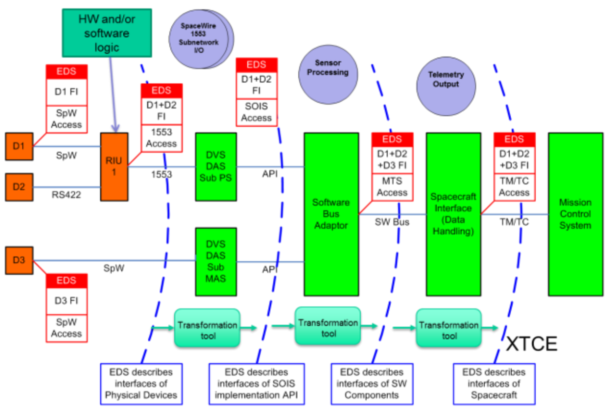

Figure 6: SEDS "Onion" Layer Diagram

Layers of the Onion

A SEDS is applicable at several layers of the system architecture from the hardware device on a MIL 1553, or Ethernet bus, through a device concentrator hub, up to the software components exchanging messages. The SEDS builds on the lower layers and propagates information up the software stack. At the hardware device layer a SEDS would contain data link, encoding, engineering unit conversions, sample rates, and interaction state machine information. For an example star tracker on a Mil1553 communications bus the manufacturer's SEDS would contain entries specifying that word offset 4 of SubAddress 02 contains a 16 bit signed integer with a mx+b linear conversion of 0.1 * value + 3.0 to calculate degrees Celsius and that the value can be read 100milliseconds after a sample command is written to SubAddress 4 word 0. That is sufficient specification to write software to retrieve the value within a specific software architecture. This information can be used directly by a tool to generate the device data access code as has been successfully demonstrated by ESA study projects. The device SEDS would also contain additional meta-data specifying items such as the operating temperature range of the device or other operational limits.

As we move up the software stack or "onion", the message layer SEDS (Software Bus Adapter) would be created from the device layer SEDS with additional information added such as the message format and new layout of the engineering unit converted parameters as well as the mnemonic name of the parameter. It would also retain limits and other meta-data associated with the parameter that would benefit the layers above. If no further data transformations were needed the message layer SEDS could be used directly to create the ground system command and telemetry database. In the "onion" diagram this would be in files(s) containing another CCSDS and OMG standard, XML Telemetric & Command Exchange (XTCE).

cFS use of SEDS

Although the current cFS tool chain is only using part of the SEDS potential we are seeing significant improvements in the development process. For the message exchange layer of the cFS architecture the SEDS specifies the packet header, data layout, and encodings, as well as metadata such as limits, and mnemonics. Typing "make" to create the load files for a cFS target system will start a process where the build tools will read a cFS system configuration file, generate the "c" language header files, generate the ground system command and telemetry database, and generate documentation. Before the SEDS these were typically done by hand or custom tools. Now there is a single common definition of the component data interfaces from which other definitions are derived reducing interface errors and ensuring consistency across the systems.

Another area being prototyped for the cFS is runtime endianness conversions for data exchanged between big and little endian systems. This would enable mixed endian system to be deployed without writing custom conversion software as shown in Figure 7. This is a common problem when interfacing to ground control and support systems. Many of the radiation hardened flight systems use big endian Power PC processors but ground systems use lower cost x86 little endian processors. Without a standard data exchange definitions projects would incur the expense of writing custom code to perform the data type transformations.

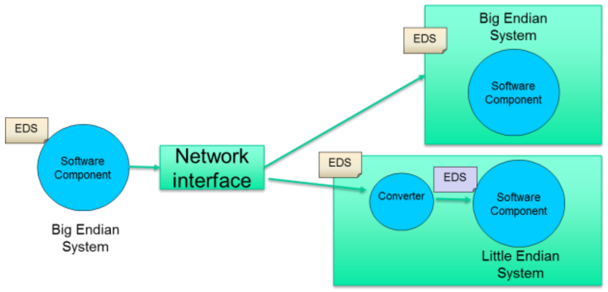

Figure 7: SEDS Mixed Endian Use Case

Conclusions

With the addition of the SOIS EDS, the combination of the flight proven cFS, and existing CCSDS standards provide a software platform and tools for organizations to rapidly develop, operate and maintain spacecraft and embedded systems at a much lower cost. Using an open software architecture and open international standards enables collaboration and interoperability across organizations no matter where they located. NASA has already seen benefits and is targeting SEDS integration in an upcoming open release of the cFS. ESA is moving forward with projects to mature the tools for use on upcoming missions. With the standards adoption by space agencies, academic and commercial satellite developers we expect component manufacturers to follow. This will bring about the full promise of the standard. Select a device or software component from the library/catalog and auto-generate much of the software interfaces, models, tests, ground control system databases, and simulations, easing integration and lowering mission development and operations costs.

Acknowledgments

The author would like to acknowledge the CCSDS SOIS team: Richard Melvin, United Kingdom Space Agency (UKSA), Ramon Krosely, Air Force Research Lab (AFRL), Glenn Rakow - NASA Goddard Space Flight Center, Marek Prochazka, European Space Agency (ESA), Xiongwen He - China Academy of Space Technology (CAST), Joseph Hickey, NASA Glenn Research Center, Andrew Loveless, NASA Johnson Space Center, Kevin Rice, NASA Goddard Space Flight Center, and Dr James Lyke and his team for their pioneering work at AFRL on the SPA architecture.

References:

1. Further information on cFS available at https://cfs.gsfc.nasa.gov/
2. Further information on CCSDS available at www.ccsds.org
3. 876.0-R-0 XML Specification for Dictionary of Terms for Electronic Data Sheets, CCSDS, April 2016.
4. 876.1-R-2 Dictionary of Terms for Electronic Data Sheets, CCSDS, June 2016.
5. Lyke, J. et al. “Lessons Learned: Our Decade in Play-and-play for Spacecraft” SSC14-V-1, 28th Annual AIAA/USU Conference on Small Satellites

Acronym List

AFRL	Air Force Research Lab
CCSDS	Consultative Committee for Space Data Systems
cFS	Core Flight System
EDS	Electronic Data Sheet
ESA	European Space Agency
GNU	GNU's Not Unix
GPM	Global Precipitation Measurement
LRO	Lunar Reconnaissance Orbiter
MMS	Magnetospheric Multiscale (mission)
NASA	National Aeronautics and Space Administration
NICER	Neutron star Interior Composition Explorer
OMG	Object Management Group
PACE	Plankton, Aerosol, Cloud, ocean Ecosystem
SEDS	SOIS Electronic Data Sheet
SOIS	Spacecraft Onboard Interfaces Services
SPA	Space Plug and Play Avionics
UAV	unmanned aerial vehicle
WFIRST	Wide Field InfraRed Survey Telescope
XML	eXtensible Markup Language
XTCE	XML Telemetric & Command Exchange
xTEDS	extensible transducer electronic data sheets